

Projet GPGPU

Automne 2025 - GISTRE/SCIA S9

Objectif

Plugin GStreamer de séparation fond / objets mobiles dans des vidéos

INPUT = flux vidéo (stabilisé) / OUTPUT = vidéo avec différences en rouge

Étape préliminaire dans de nombreuses chaînes de traitement

Beaucoup d'opération locales \Rightarrow Bon candidat pour une optimisation GPU



Frame t



Frame $t+1$



Différence ($t, t+1$)

Ce que vous devez savoir sur GStreamer

1. C'est pénible à faire fonctionner
2. C'est très efficace
3. Vous n'aurez qu'à regarder le contenu de la fonction

```
void compute_cu(ImageView<rgb8> in)
```

`compute_cu`

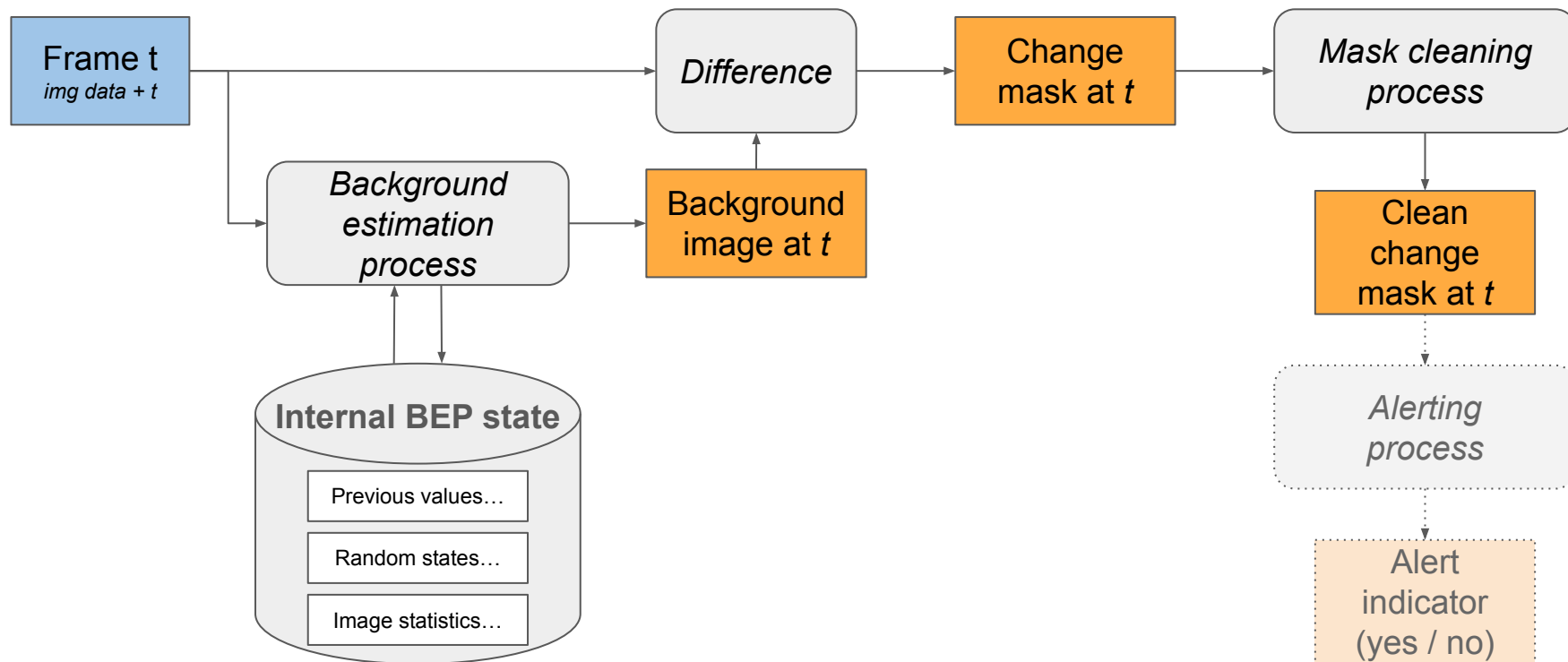
- est appelée pour chaque nouvelle frame
- `in` contient les données de la frame courante
- Vous devez utiliser `in` pour mettre à jour l'état interne de votre système
→ *variables statiques pour assurer une persistance entre frames*
- Vous devez modifier `in` pour renvoyer la frame qui sera affichée ou sauvegardée (architecture “zero copy”)

Input
data

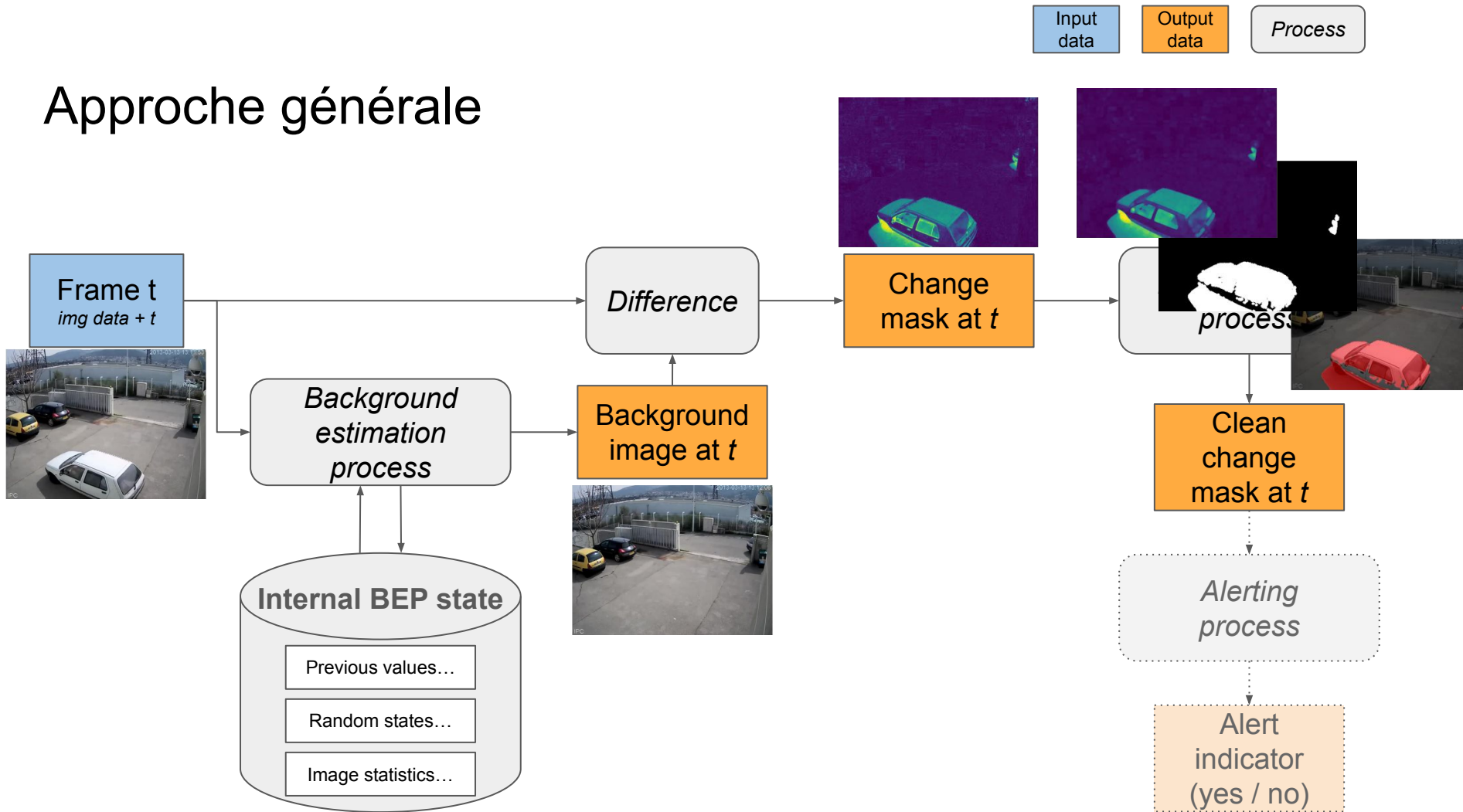
Output
data

Process

Approche générale



Approche générale



Détail des étapes

1. Background estimation process

Pixel-wise iterative process, for each frame

(For each pixel position p)

States: rs = K reservoir values and weights, $randState$ = 1 $rand$ state

Init: $rs[i].w = 0$ for all i in $0..K-1$ (performed once for all frames) ($rs[i].rgb$ may be random or $(0,0,0)$)

// find matching reservoir

$m_idx = \text{find_matching_reservoir}(p.rgb, rs)$

// update weights and samples

if $m_idx \neq -1$ && $rs[m_idx].w > 0$: **// matching**

$rs[m_idx].w += 1$

$rs[m_idx].rgb = ((rs[m_idx].w - 1) * rs[m_idx].rgb + p.rgb) / rs[m_idx].w$

else if $m_idx \neq -1$ && $rs[m_idx].w == 0$: **// empty slot**

$rs[m_idx].rgb = p.rgb$

$rs[m_idx].w = 1$

else: **// no match and no empty slot, perform weighted reservoir replacement**

min_idx = index of the reservoir with the smallest weight

$total_weights$ = sum of $rs[i].w$ for i in $0..K-1$

if $(randfloat * total_weights \geq rs[min_idx].w)$: replace $rs[min_idx].rgb$, set w to 1

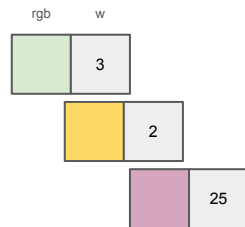
// then cap weights to $MAX_WEIGHTS$

...

// and set the background (value of the pixel to return) to the rgb with max weight

...

Last samples and weights
for each pixel position



↑ alignment optimisation possible

Based on “weighted reservoir sampling”, an online sampling algorithm.

m_idx = find_matching_reservoir(p, rs)

```
m_idx = -1;
for every reservoir rs[i] in rs:
    if rs[i].w > 0:
        if |p.rgb.r - rs[i].rgb.r| < RGB_DIFF_THRESHOLD &&
           // same for g and b
           m_idx = i;
           break; // equivalent to return i;
    else:
        m_idx = i;

return m_idx;
```

→ find first matching reservoir or empty reservoir if no match, or -1 if no empty reservoir



Be careful when computing the difference with unsigned integers: cast them to signed integer first!

Handling random number generators

We need to maintain a random state for each pixel position.

```
static Image<curandState> rng states;
```

These states are initialized using a kernel which calls

```
curand_init(seed, global_pixel_pos, 0, &randState_row[x]);
```

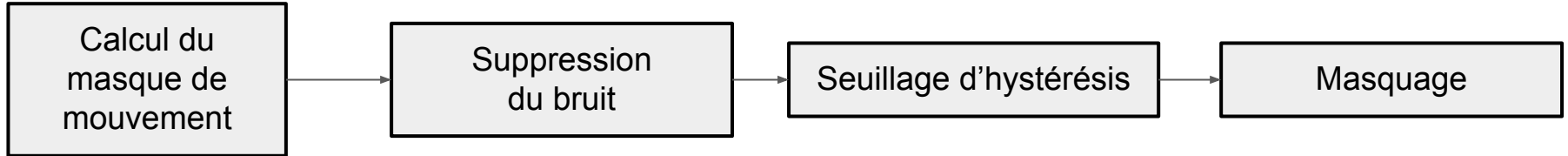
for each position.

These states are updated when later drawing a random number using

```
float rand_val = curand_uniform(&randState_row[x]);
```

($0 \leq \text{rand_val} \leq 1$)

2. Mask cleaning process overview



Résultat du processus précédent.

C'est une carte qui indique pour chaque pixel un score de mouvement ≥ 0 .



Ouverture morphologique par un disque de rayon 3.

La taille du rayon doit être ajusté à la taille de l'image.



*Seuillage d'hystérésis.
Seuil bas: 4
Seuil haut: 30*

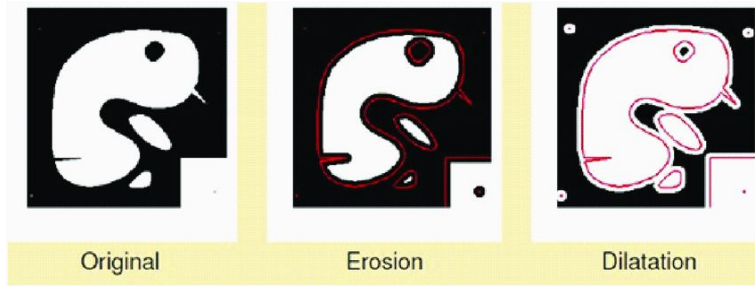
La valeur de seuil dépend de l'espace colorimétrique.



*input + 0.5 * red * masque*

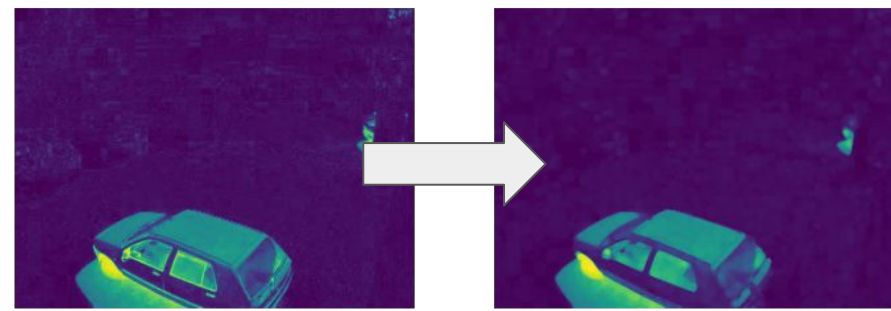
2.1 Suppression du bruit

Ouverture morphologique (érosion suivi d'une dilatation)



https://en.wikipedia.org/wiki/Mathematical_morphology

[https://en.wikipedia.org/wiki/Opening_\(morphology\)](https://en.wikipedia.org/wiki/Opening_(morphology))



Erosion:

new value for $p(x,y)$ = **min** value in neighborhood of $p(x,y)$

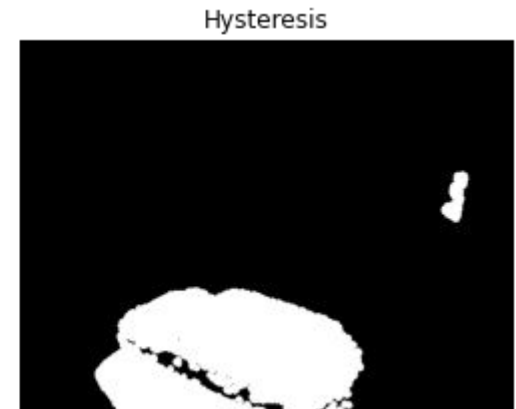
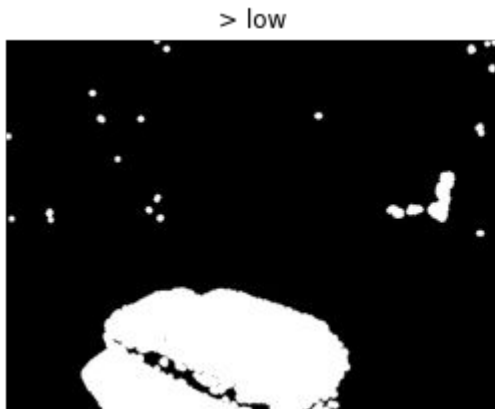
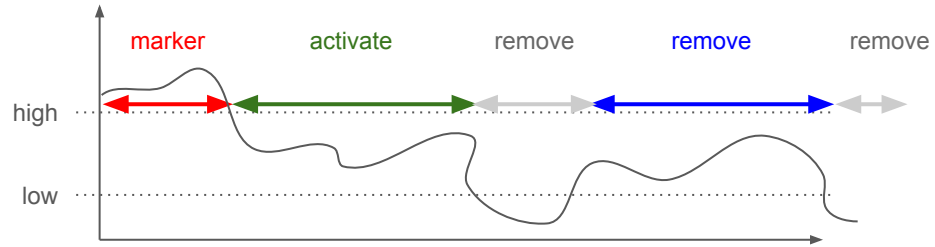
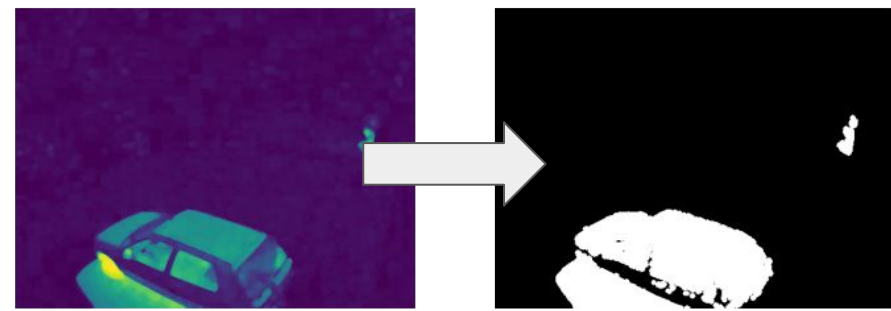
Dilation:

new value for $p(x,y)$ = **max** value in neighborhood of $p(x,y)$

2.2. Seuillage d'hystérésis

Principe :

- Supprimer les signaux faibles
- Propager les signaux forts vers les signaux moyens



2.2. Implémentation de la reconstruction d'hystérésis

Idée:

On propage les pixels des marqueurs dans le masque jusqu'à stabilité.

Les marqueurs sont initialisés avec les éléments supérieurs au seuil haut.

L'entrée contient tous les éléments supérieurs au seuil bas.

```
__device__ bool has_changed;

__global__ void reconstruction(input, marker, out) {
    int p = .... + threadIdx.x;
    if (out[p] || !input[p]) // already processed or too low
        return;
    if (marker[p]) { // initialize output with markers
        out[p] = true;
        has_changed = true;
        return;
    }
    for (int q : neighbors(p)) // Is any of my neighbors active?
        if (out[q]) { // propagate neighbor activation to me
            out[p] = true;
            has_changed = true;
        }
    }
}

int main() {
    out = black_image;
    has_changed = false;
    while (has_changed)
        reconstruction<<<>>>(input, marker, out);
}
```

Matériel fourni

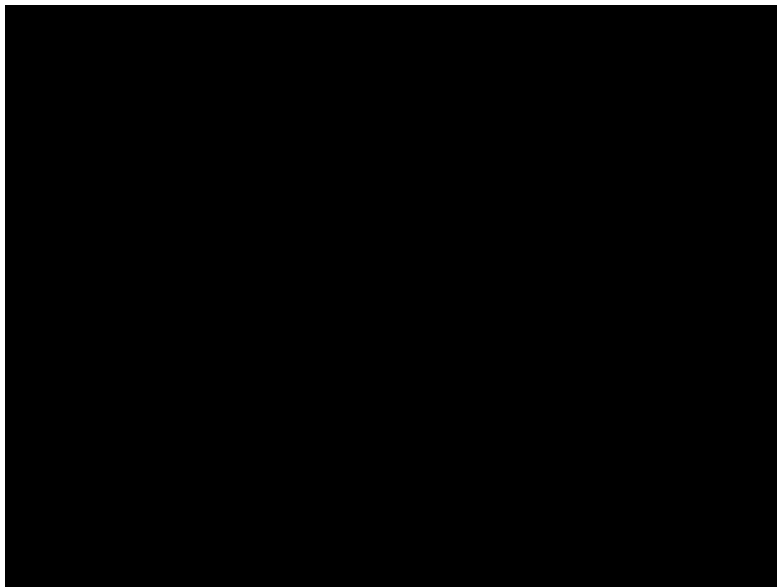
Code gstreamer

- Code de base fourni sur Moodle
- Vous devez implémenter un filtre GStreamer CUDA et CPP

Idéalement, en intégrant les paramètres suivants :

```
bg=uri           : uri vers une image de fond (default="" => estimé)
opening_size=(int) : taille de l'ouverture
th_low=(int)      : valeur basse du filtre (default=3)
th_high=(int)     : valeur haute du filtre (default=30)
```

```
bg_sampling_rate=(int) : intervalle d'échantillonnage des frames pour l'estimation de fond
                        (default=500ms)
bg_number_frame=(int)  : nombre de frames utilisées pour l'estimation de fond (default=10)
```



```
gst-launch-1.0 uridecodebin uri=file:///tmp/sintel_trailer-480p.webm ! videoconvert ! "video/x-raw,  
format=(string)RGB" ! cudafilter ! videoconvert ! video/x-raw, format=I420 ! x264enc ! mp4mux ! filesink  
location=video.mp4
```

Demo

Attendus

Critères d'évaluation

1. **Code correct** \Rightarrow résultats ACCEPTABLES au niveau qualitatif
Avec cette méthode, les résultats ne seront pas optimaux
2. **Vitesse** \Rightarrow plus le framerate est rapide, mieux c'est.

Conseils

1. **Avoir une version C++ fonctionnelle** \Rightarrow baseline
2. **Git tag des versions du programme** \Rightarrow permet de mesurer plusieurs versions/optimisations du programme (+ conserver variantes pour comparaison)
3. **Faire les optimisations une par une** \Rightarrow permet d'évaluer les optimisations réellement significatives de façon individuelle

Livrables

1. Implémentation

- Source code for C++ CPU reference
- Source code for CUDA implementation(s)
- Source code for benchmark tools
- Build scripts (GNU Make, CMake...)

Nous devons être capables de reproduire vos résultats.

2. Rapport succinct

- Description du sujet (choix pour chaque étape, ce qui a été implémenté sur GPU)
- Répartition des tâches par membre du groupe
- Benchmarks et graphiques des performances des versions (CPU + GPU + GPU Optimisé ##)
- Analyse des performances et des bottlenecks (graphiques nsight / nvprof)

3. Slides de la soutenance

4. Répartition des groupes (sur Moodle) => aujourd'hui à la fin du TP (pour les affectations de soutenance)

Soutenances

18 et 19 déc

- 15' présentation
- 5' démo

→ *Données:*

https://drive.google.com/drive/folders/1pAisl_mfdfXrkEqK-8xU-gblRENLS85I

- 5' discussion

Soutenances sur Teams. Les liens vous seront envoyés la semaine précédente.

Projet par groupe de 4.

Tous les membres du groupe doivent être présents à la soutenance.

Vous devez rendre **tous** les fichiers **le 17 déc. au soir.**